Monitoring the service-based system lifecycle with SALMon

Marc Oriol*, Xavier Franch, Jordi Marco
Universitat Politècnica de Catalunya – BarcelonaTech, ESSI – UPC. 08034 Barcelona, c/Jordi Girona 1-3, Spain
{moriol, franch}@essi.upc.edu, jmarco@lsi.upc.edu

* Corresponding author. Address: Universitat Politècnica de Catalunya – BarcelonaTech, ESSI – UPC. 08034 Barcelona, c/Jordi Girona 1-3, Omega Building S-206, Spain.
e-mail: moriol@essi.upc.edu Telf: +34 93 413 78 62.

Abstract

[Context and motivation] Service-Based Systems are highly dynamic software systems composed of several web services. In contrast to other types of systems, Service-Based Systems rely on service providers to ensure that their web services comply with the agreed Quality of Service. Delivering an adequate Quality of Service is a critical and significant challenge that requires monitoring along the different activities in the Service-Based System's lifecycle.
[Question/problem] Current monitoring systems are designed to support specific activities (e.g. service selection, adaptation, etc.), but do not fulfil the requirements of all the activities in the Service-Based System's lifecycle.
[Principal ideas/results] In this paper, we present SALMon, a QoS monitoring framework able to support the whole Service-Based System's lifecycle. SALMon is highly versatile, since it combines different strategies for its configuration (model-based and invocation-based) and for the way it gets the Quality of Service (passive monitoring and online testing). Furthermore, its architecture supports easy extensibility with new quality attributes, independence of the technology of the monitored services and interoperability with other tools. We conducted a performance evaluation over real web services using suitable estimators for response time and evaluated both its overhead and capacity.
[Contribution] SALMon provides infrastructure that can be used in very different scenarios, as exemplified in this paper, both in terms of the lifecycle's phase addressed and the type of system (pure Service-Oriented Architecture, cloud-based systems, etc.). This diversity of situations addressed makes SALMon a significant contribution both for practitioners that may be interested in integrating a working technology in their software solutions, and for researchers who can conduct their investigation on top of a reliable infrastructure.

Keywords: Service Based System, Monitoring, Quality of Service, QoS, Web Service

1. Introduction

Service-Based Systems (SBS) are highly dynamic software systems composed of several web services coming from different, possibly heterogeneous providers (Andrikopoulos, Bertoli, & Bindelli, 2008). They follow the principles of the Service Oriented Architecture (SOA), which improves the reusability, abstraction, loose coupling and high cohesion of the web services that compose them (Erl, 2008). Due to these characteristics, SOA has been established as a successful paradigm in both industry and academia. In industry, SOA has been proven to be a suitable solution for many projects in the last decade

(Bieberstein, Laird, Jones, & Mitra, 2008), whereas in the academia it is acknowledged to be a consolidated approach which progresses along with cloud computing and the future internet (Papazoglou, Pohl, Parkin, & Metzger, 2010).

The adoption of an SBS-based approach to the system's development impacts on its lifecycle, with additional stages and activities that do not occur in other paradigms. Fig. 1 presents a possible lifecycle based on the SBS lifecycle proposed in (Papazoglou et al., 2010) complemented with the SBS lifecycle as presented in (Gu & Lago, 2007).

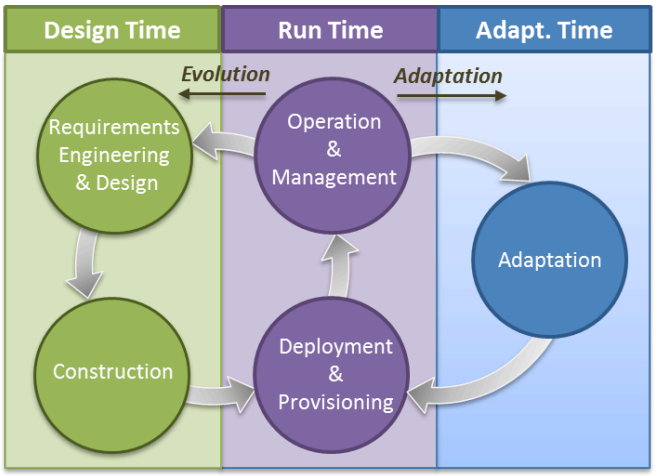

Fig. 1 Lifecycle of an SBS.

In contrast to other traditional software systems, the dynamic behavior of SBS requires up-to-date Quality of Service (QoS) information for their proper management in the different stages of the SBS lifecycle shown, from their initial construction until their decommission. The QoS that the web service must accomplish is usually expressed in the form of a contract between the service client and the service provider known as Service Level Agreement (SLA). Table 1 summarizes the activities in which the different lifecycle stages make use of QoS information.

Table 1: Activities that require up-to-date QoS information.

Stage	Activity
Requirements Engineering & Design	Does not require QoS information but instead defines the QoS attributes that will be needed to check the fulfilment of the adopted SLA, influencing then the other four stages
Construction	*Service selection*: requires QoS information in order to select the most suitable service from a set of candidate services to fulfil a certain task.
Deployment & Provisioning	*Service deployment*: requires QoS information in order to deploy the services which are managed by the same SBS developer under the proper infrastructure.
Operation & Management	*Quality assessment*: requires QoS information to assure at execution time that the conditions defined in the SLA are met.
Adaptation	*Service adaptation*: requires QoS information to correctly select and execute the adaptation strategy when a violation of the service level occurs (or is expected to occur).

Providing such QoS information has resulted in different technological solutions built around a **monitor**. "A monitor is a program or set of programs that observes the execution of the SBS […] and is defined and implemented on the basis of a monitoring specification that describes the properties and events to be observed" (S-Cube, 2009). QoS is one of such properties.

There are several existing solutions that are well fit for some of the activities listed in Table 1, but as our state of the art will show later, none of them have been used for supporting all of them altogether. Under these circumstances, these activities can only be successfully accomplished by combining two or more different monitors in the same SBS, leading to redundancies and affecting negatively the performance of the system. Furthermore, some compatibility problems between monitors might arise (e.g. inconsistent approaches, different interfaces, incompatible dependencies, etc.).

As an illustrative example, let's consider an SBS that has a monitor capable of supporting **quality assessment.** In this activity, the monitor is used to assess that the availability of the service is over 95% during the year. However, if the service becomes unavailable, whilst the SLA is still met, the same monitor might be unable to **identify the adaptation need**. This is the case when monitors are highly coupled to the conditions and can't distinguish the implications of the violation, or with which components it should interact with. Under this scenario, another monitor shall be installed to take care on the adaptations needed, leading to the problems mentioned.

Moreover, installing a list of redundant monitors makes the maintainability of the SBS more complex and prone to errors. For instance, if an adaptation on the SBS is performed: if a service A is replaced by a service B, all the monitors and the monitoring rules should be ported from A to B.

To fill this gap, in this paper we present SALMon, a versatile service monitoring framework that provides the QoS information required by the different activities previously mentioned. In order to satisfactorily fulfil the diverse requirements of the different activities involved, the monitoring framework provides different strategies to get the QoS by combining passive monitoring and online testing strategies, supporting different deployment configurations, the ability to extend the monitoring solution with new quality attributes and their metrics, as well as offering an open architecture to deal with web services developed and running in different technologies.

The remainder of the paper is structured as follows: In Section 2, we identify the monitoring requirements of the different activities in the SBS lifecycle. In Section 3, we present the related work and how do they cover such requirements. In Section 4, the SALMon monitoring framework is presented. In Section 5, we describe the usage of SALMon in the different activities. In Section 6, the performance evaluation is presented. Finally, in Section 7, we present the conclusions and future work.

2. Monitoring requirements

Designing a comprehensive monitoring solution requires a careful analysis of the needs posed by the four activities presented in Table 1. This section presents the list of requirements derived from these needs. We obtained these requirements through the following elicitation techniques: on the one hand, we revised the literature in the fields of service selection (Kang, Liu, Tang, Liu, & Fletcher, 2011)(S. Wang, Sun, & Yang, 2012)(Michlmayr, Rosenberg, Leitner, & Dustdar, 2010), service deployment (Gentzsch et al., 2013)(Marosi, Kecskemeti, Kertesz, & Kacsuk, 2011), quality assessment (Shu & Meina, 2010) and service adaptation (Di Nitto, Ghezzi, Metzger, Papazoglou, & Pohl, 2008)(Baresi, Guinea, Pistore, & Trainotti, 2009)(Guinea & Kecskemeti, 2011) to identify the common requirements for these activities. On the other hand, we conducted meetings and interviews with members of different research groups who work on these activities[1]. Such requirements were later validated through the implementation of a monitoring solution that fulfilled the needs of these research groups for service selection (Cabrera et al., 2011)(Cabrera et al., 2009), service deployment (Kertesz et al., 2012a)(Kertesz et al., 2012b) (Kertesz et al., 2013), quality assessment (Muller et al., 2012) (Muller et al., 2013) and

[1] These collaborations were made in the context of the S-Cube FP7 Network of Excellence, http://www.s-cube-network.eu/.

service adaptation (Oriol, Qureshi, Franch, Perini, & Marco, 2012) (Schmieders, Micsik, & Oriol, 2011) (Sammodi et al., 2011) (Franch et al., 2011) (Burgstaller et al., 2009).

Firstly, we present the high-level requirements that are common along the different activities, and hence apply to the whole service lifecycle; and then we describe the specific requirements for the particular needs of each activity.

Core requirements along the SBS lifecycle: SBS may incorporate web services implemented in different languages (e.g. Java, BPEL, .NET, etc.) and executed in different engines (e.g. Axis2, Websphere, Glassfish, etc.), which might interact with different messaging protocols (e.g. SOAP, REST), leading to an heterogeneous system combining different technologies (Zheng, Zhang, & Lyu, 2010). Hence, the monitor shall be capable of monitoring different types of web services, regardless of their technology or infrastructure details (R1.1). As SBS include different web services for different purposes, the list of quality attributes the monitor is able to handle shall be extensible as to meet the end user's needs, including domain specific quality attributes (O Moser, Rosenberg, & Dustdar, 2012) (R1.2). On the other hand, requirements and quality attributes to monitor might change, new web services may emerge, adaptations might be performed, etc. The monitor shall be dynamically reconfigurable to cope with such dynamicity of an SBS (R1.3). Finally, the monitor should be easily interoperable with the different tools that support the aforementioned activities in the SBS lifecycle (R1.4).

Requirements on service selection: Service selection frameworks discover and rank web services registered in a repository. These frameworks provide a ranked list of web services that fulfil not only the functional requirements of the users, but also their non-functional requirements. These non-functional requirements are expressed in terms of conditions over quality metrics (e.g. "response time < 200 ms. AND availability > 90%").

To obtain the values of such quality metrics, two approaches have been proposed in the literature. On the one hand, some approaches require a distributed monitoring system in which the services are being monitored during its execution (Kang et al., 2011)(S. Wang et al., 2012) (R2.1). On the other hand, in other approaches, it is the same framework that requires a monitor to execute periodically the service to obtain the QoS (Michlmayr et al., 2010)(Cabrera et al., 2011) (R2.2).

Requirements on SBS deployment: The QoS of the SBS and its web services is strongly affected by the QoS of the underlying infrastructure where they are deployed (Di Nitto et al., 2008). The deployment of an SBS includes the composition, internal web services and their resources. Cloud computing is emerging as a leading solution to deploy an SBS. According to Gartner, by 2017, over 50% of large Software as Service (SaaS) providers will offer an integrated Platform as a Service (PaaS) in the cloud (Michell Smith et al., 2013). In the field of cloud computing, there are frameworks that select the cloud that better fulfils the requirements as well as allocating the resources for the SBS to be deployed (Gentzsch et al., 2013)(Marosi et al., 2011). As a primary requirement, these frameworks require monitoring solutions to retrieve the QoS at the infrastructure level (Kertesz et al., 2013) (Kertesz et al., 2012b) (Kertesz et al., 2012a) (R3.1).

On the other hand, as the cost of PaaS is usually linked to its usage and consumption, the monitor is required to minimize the consumption of resources to lower operating costs (Zhang, Cheng, & Boutaba, 2010)(Kertesz et al., 2013) (R3.2).

Requirements on quality assessment: During the execution of the SBS, the QoS of the constituent web services is dynamic and highly changing. The primary goal for monitoring in this phase is to ensure that the dynamic quality attributes meets at runtime the agreed SLA, and hence, being able to configure automatically the monitor from SLAs is a key aspect in this activity (Muller et al., 2013) (R4.1). Moreover, the monitor shall be able to recognize different SLA notations (Muller et al., 2013) (R4.2).

Since the QoS stated in the SLA refers explicitly to the interaction between the service client and the service, the monitored QoS shall be from the real execution of the involved parties (Shu & Meina, 2010) (R4.3).

Requirements on SBS adaptation: In self-adaptive SBS, when the QoS of a web service doesn't meet the required level objectives, an adaptation action is triggered to restore the QoS. We distinguish between proactive adaptation (the system adapts before the malfunction occurs) and reactive adaptation (the system adapts when the malfunction has already occurred). Proactive adaptation is usually accomplished by using a monitor that periodically executes the service to identify any violation before the user experiences the malfunction (Di Nitto et al., 2008) (R5.1), whereas reactive adaptation is accomplished by monitoring the real execution of the service client (Baresi et al., 2009)(Guinea & Kecskemeti, 2011) (R5.2).

A brief summary of these requirements is depicted in Table 2. It must be taken into account that the list is not intended to provide a comprehensive set of requirements for particular solutions, as each one differs on their own specific needs. Instead, the list provides a framework of understanding in the main requirements that a monitor should achieve to support each activity.

Table 2: List of monitoring requirements.

Requirement	
1. Along the lifecycle	
R1.1	The monitor shall be capable of monitoring different types of web services, regardless of their technology or infrastructure details.
R1.2	The monitor shall be extensible to monitor new quality attributes.
R1.3	The monitor shall be dynamically configurable.
R1.4	The monitor shall be interoperable with the different required components to support the activity of the SBS lifecycle.
2. Service selection	
R2.1	The monitor shall be able to passively monitor services in a distributed environment.
R2.2	The monitor shall be able to actively test services periodically.
3. SBS Deployment	
R3.1	The monitor shall retrieve the values of quality attributes at the infrastructure level.
R3.2	The monitor shall minimize the number of resources consumed.
4. Quality Assurance	
R4.1	The monitor shall be automatically configurable from SLAs.
R4.2	The monitor shall be able to recognize different SLA notations
R4.3	The monitor shall retrieve the measurements from the real usage of web service clients.
5. SBS adaptation	
R5.1	The monitor shall be able to identify QoS violations proactively.
R5.2	The monitor shall be able to identify QoS violations reactively.

3. Related Work

To develop the Related Work, we have followed the principles and guidelines of Systematic Literature Reviews (SLRs) as defined by Kitchenham (Kitchenham & Charters, 2007). Nevertheless, the goal in this paper is not to develop an exhaustive SLR with all the work available in the literature, but to report in a systematic manner the list of relevant contributions similar to our work focusing on the quality (as sustained by the publishing venues) rather than the quantity of papers. The selected papers are then evaluated in respect of the requirements elicited in the previous section.

3.1 Selection of papers

We have performed a manual search from 2008 to 2014 on the following list of journals and conferences:

Journals: ACM-TOSEM, ACM-TWEB, Elsevier-AES, Elsevier-IST, Elsevier-JSS, Elsevier-ESWA, IEEE-Computer, IEEE-Internet Computing, IEEE-Software, IEEE-TSC, IEEE-TSE, IGI global-IJWSR, Springer-ASE, Springer-SQJ, Springer-WWW.

Conferences: ASE, CAiSE, FSE, ICSE, ICSOC, ICWS, SCC, WISE, WWW

The list of journals was obtained from the top ranked journals in services, software engineering and information systems engineering based on their JCR impact factor. Similarly the list of conferences was obtained from the top ranked conferences in the same areas based on the CORE index[2], selecting those which have a CORE-A status.

We have performed the search with the terms "monitor* AND service*" limiting the search results to the last 5 years, gathering hence the most updated monitoring solutions with respect to the latest standards. The terms have been applied to title, abstract and keywords. By applying this search protocol, we found 292 papers covering the search criteria. 113 papers were discarded by title, 125 by abstract, and 37 papers were discarded after a fast reading, leading to a total of 18 papers that present 14 different approaches. Details of the search and filtering process can be checked at (Oriol, Franch, & Marco, 2014).

3.2 Analysis

In this subsection, we analyse to which degree the retrieved papers fulfil the requirements identified in Section 2, and subsequently their suitability to support the different activities of the SBS lifecycle.

For each proposal, we have evaluated the fulfilment of each requirement. The different possible values can be either satisfied (tick figure), unsatisfied (cross figure), or in the case it was not clearly described, unknown (question mark). We have also checked if the monitoring framework has been validated through the inclusion of the monitor in each of the activities presented (depicted with the same symbols in the rows labelled as 'Val.'). The results are summarized in Table 3.

Requirements along the lifecycle: Most of the presented approaches are limited to specific web service technologies and cannot deal with any type of service (R1.1). Most approaches are limited to service compositions written in WS-BPEL (Oliver Moser et al., 2008)(Oliver Moser et al., 2010)(Baresi et al., 2010)(Baresi & Guinea, 2011)(Baresi et al., 2009)(Guinea & Kecskemeti, 2011), SOAP-based web services (Fei et al., 2008)(Mahbub et al., 2010)(Benharref et al., 2009), RESTful (Katsaros et al., 2011), or services implemented in Axis/Axis2 (Q. Wang et al., 2009)(Raimondi et al., 2008). These technological limitations constrain the applicability of the monitors to a narrow set of scenarios. On the contrary, there are a few approaches that provide a solution not attached to a particular technology (Lin et al., 2009)(Comuzzi et al., 2009)(Ortiz & Bordbar, 2009)(Psiuk et al., 2012)(Alhamazani et al., 2014).Regarding the quality attributes, most of the approaches are extensible to monitor new quality attributes (R1.2) with a few exceptions. Ortiz et al. (Ortiz & Bordbar, 2009) is only able to monitor a set of predefined metrics and lacks of important ones (e.g. availability) and Raimondi et al. (Raimondi et al., 2008) can only monitor time-related metrics. Finally, most of the monitors are dynamically configurable (R1.3), with the exception of Benharref (Benharref et al., 2009) and Raimondi (Raimondi et al., 2008), which cannot be reconfigured at runtime.

[2] Available at http://core.edu.au/index.php/categories/conference%20rankings

Table 3: Fullfilment of the requirements in the related work

	O. Moser (Oliver Moser, Rosenberg, & Dustdar, 2008)(Oliver Moser, Rosenberg, & Dustdar, 2010)	**SECMOL-WSCol**(Baresi, Guinea, Nano, & Spanoudakis, 2010) (Baresi & Guinea, 2011)	**Dynamo&Astro** (Baresi et al., 2009)(Guinea & Kecskemeti, 2011)	**K. Lin** (Lin, Panahi, Zhang, Zhang, & Chang, 2009)	**M. Comuzzi** (Comuzzi, Kotsokalis, Spanoudakis, & Yahyapour, 2009)	**G. Ortiz** (Ortiz & Bordbar, 2009)	**L. Fei** (Fei, Fangchun, Kai, & Sen, 2008)	**K. Mahbub** (Mahbub, Spanoudakis, & Zisman, 2010)	**M. Psiuk** (Psiuk, Bujok, & Zieliski, 2012)	**Q. Wang** (Q. Wang et al., 2009)	**A. Benharref**(Benharref, Dssouli, Serhani, & Glitho, 2009)	**F. Raimondi** (Raimondi, Skene, & Emmerich, 2008)	**G. Katsaros** (Katsaros, Kübert, & Gallizo, 2011)	**CLAMS** (Alhamazani et al., 2014)	**SALMon**
	1. Along the lifecycle														
R1.1	×	×	×	✓	✓	✓	×	×	✓	×	×	×	×	✓	✓
R1.2	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	×	✓	×	✓
R1.3	✓	✓	✓	✓	✓	✓	✓	?	✓	✓	×	×	✓	?	✓
R1.4	*They have been proven compatible with their respective frameworks*														✓
	2. Service selection														
R2.1	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	✓	✓
R2.2	×	✓	✓	×	×	×	×	✓	✓	✓	×	×	✓	✓	✓
Val.	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	✓
	3. SBS Deployment														
R3.1	×	×	✓	✓	✓	×	×	×	×	✓	×	×	✓	✓	✓
R3.2	✓	×	×	×	×	×	✓	×	✓	×	×	×	×	?	✓
Val.	×	×	✓	✓	×	×	×	×	×	×	×	×	✓	✓	✓
	4. Quality Assurance														
R4.1	×	✓	×	×	✓	×	×	×	×	×	×	✓	×	×	✓
R4.2	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓
R4.3	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	✓	✓
Val.	×	✓	×	×	✓	✓	×	×	✓	✓	✓	✓	×	×	✓
	5. SBS adaptation														
R5.1	×	✓	✓	×	×	×	×	✓	✓	✓	×	×	✓	✓	✓
R5.2	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	✓	✓
Val.	✓	✓	✓	✓	×	×	✓	✓	×	✓	×	×	×	×	✓

Service selection: All of the aforementioned monitoring frameworks can monitor the services either passively (R2.1), or actively using online testing (R2.2). However, only SECMOL-WSCol (Baresi et al., 2010)(Baresi & Guinea, 2011), Dynamo&Astro (Baresi et al., 2009)(Guinea & Kecskemeti, 2011), M. Psiuk (Psiuk et al., 2012), Q. Wang (Q. Wang et al., 2009) and CLAMS (Alhamazani et al., 2014) are able to combine both approaches. This ability to combine the two approaches allows the monitoring framework to be used in different service selection frameworks. Nevertheless, none of these approaches has been specifically used in service selection scenarios to validate its adequacy in a service selection framework. Only Mahbub et al. (Mahbub et al., 2010) has been used in service selection, although just for SOAP-based passive monitoring.

SBS deployment: Only a few approaches are able to monitor quality attributes at the infrastructure level (R3.1). However, they either require to plug an infrastructure monitoring engine (Baresi et al., 2009)(Guinea & Kecskemeti, 2011)(Comuzzi et al., 2009)(Katsaros et al., 2011), which requires more resources, or they directly interact with the operating system commands (Lin et al., 2009)(Q. Wang et al., 2009), and hence are limited to a specific operating system. None of these approaches apply techniques to minimize the consumption of resources (R.3.2). Regarding validation, only Dynamo&Astro

(Baresi et al., 2009)(Guinea & Kecskemeti, 2011), G. Katsaros (Katsaros et al., 2011) and K. Lin (Lin et al., 2009) have been used for SBS deployment.

Quality assurance: Most of the approaches are able to monitor the QoS from the real usage of web service clients (R4.3) but only a few of them are able to automatically configure the monitor from an SLA document (R4.1). Moreover, the automated configuration of monitors presented are designed for a specific language, such as WS-Agreement (Baresi et al., 2010)(Baresi & Guinea, 2011) (Comuzzi et al., 2009) or SLAng (Raimondi et al., 2008) and are not extensible to other SLA notations (R4.2). Interestingly enough, a vast number of monitors have been validated for quality assurance (Baresi et al., 2010) (Baresi & Guinea, 2011) (Comuzzi et al., 2009) (Ortiz & Bordbar, 2009) (Psiuk et al., 2012) (Q. Wang et al., 2009) (Benharref et al., 2009) (Raimondi et al., 2008), although with the aforementioned limitations.

SBS adaptation: All of the aforementioned monitoring frameworks can either support proactive adaptation (R5.1) or reactive adaptation (R5.2). However, only SECMOL-WSCol (Baresi et al., 2010)(Baresi & Guinea, 2011), Dynamo&Astro (Baresi et al., 2009)(Guinea & Kecskemeti, 2011), Q. Wang (Q. Wang et al., 2009) and CLAMS (Alhamazani et al., 2014) are able to support both types of adaptation approaches. Most of the monitors in the literature have been integrated into an SBS adaptation framework (Oliver Moser et al., 2008)(Oliver Moser et al., 2010)(Baresi et al., 2010)(Baresi & Guinea, 2011)(Baresi et al., 2009)(Guinea & Kecskemeti, 2011)(Lin et al., 2009)(Fei et al., 2008)(Mahbub et al., 2010).

To conclude, all of the aforementioned monitoring frameworks support (partially) some of the activities, but none of them satisfactorily cover the whole SBS lifecycle, which requires a flexible approach to support the different activities in a general and extensible way. Therefore, we envision the need of a new approach that fulfils the requirements needed for all the activities that embrace the SBS lifecycle in a generic and extensible manner.

4. The SALMon monitoring framework

This section presents the insights of the framework, showing the monitoring framework designed to accomplish the requirements identified in Section 2. The framework, named SALMon[3], has been developed following an incremental and iterative approach. Nevertheless, we illustrate the results of such development in a linear way for the sake of readability.

4.1 SALMon's features

SALMon has been implemented with a set of features designed to accomplish the different requirements identified on each activity (see Table 4). We describe these features below:

Table 4: Features of SALMon and requirements fulfilled

	R1.1	R1.2	R1.3	R1.4	R2.1	R2.2	R3.1	R3.2	R4.1	R4.2	R4.3	R5.1	R5.2
F1			✓						✓	✓			
F2					✓	✓					✓	✓	✓
F3		✓					✓						
F4	✓												
F5													
F6				✓									
F7								✓					

[3] SALMon comes from SLA Monitoring with a swap of two letters to make it easier to remember.

F1. Combination of model-based and invocation-based strategies:

Quality assurance requires configuring the monitor automatically from an SLA specification (R4.1). However, other activities do not handle SLAs to configure the monitor. To this respect, the monitor must be able to combine two strategies to handle its configuration. On the one hand, it should be able to be configured automatically from an SLA, on the other hand, it should provide an API to configure the monitor directly. SALMon combines both approaches seamlessly.

Considering automatic configuration through SLAs, the current SLA standard is WS-Agreement (Andrieux et al., 2004). However, WS-Agreement just provides a general-purpose schema that must be extended with internal sublanguages, which leads to different WS-Agreement compliant notations (Muller et al., 2013), and therefore a mechanism able to handle these different WS-Agreement notations is required.

To address this challenge, we use model-based strategies, and in particular model transformations, to obtain, from an arbitrary SLA written in any language, a unified monitoring configuration model able to configure the monitor. We propose to use a specific type of document, the Monitoring Management Document (MMD), which includes the required information to configure the monitor. By decoupling the monitor from an SLA, the same monitor can be used to monitor different SLAs in different notations (See Fig. 2). Details of the MMD can be checked at (Muller et al., 2013).

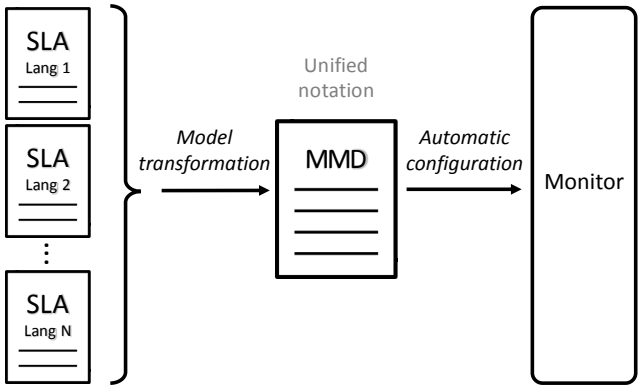

Fig. 2 Automatic configuration of monitors from diferent SLA notations.

Considering the configuration of the monitoring without SLAs, an invocation-based approach is followed. We provide an API in order to set the services, methods and metrics as well as other parameters to configure the monitor. The API is a WSDL interface, which can be invoked remotely by the client using any programming language. A high-level representation of an extract of the WSDL interface is depicted in Fig. 3.

Regardless of the adopted strategy, SALMon can dynamically be reconfigured to adapt to changes in both the SBS and the SLA (e.g. add/remove services, metrics, etc.) (R1.3).

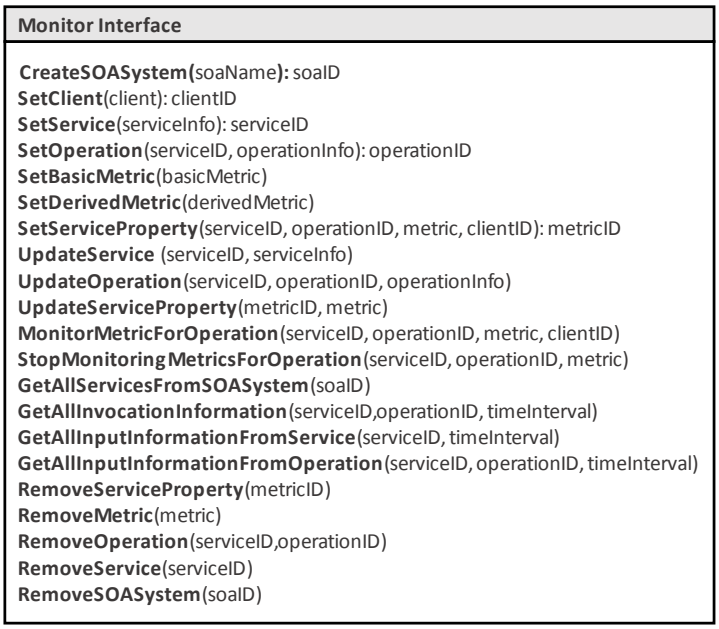

Fig. 3: Monitor interface

F2. Combination of passive monitoring and online testing strategies:

Passive monitoring consists on gathering the QoS information from the interaction between the web service and the service client. On the other hand, on-line testing consists on invoking periodically the web service to obtain the QoS information. Depending on the activity, it is required to use one passive monitoring (R2.1, R5.2) or online testing (R2.2, R5.1) to gather the QoS.

In order to satisfy the needs of the different activities, we combine both online testing and passive monitoring strategies in the same framework. To do so, the tester uses exactly the same monitoring infrastructure as an end-user of the service (see Fig. 4).

Both strategies are not mutually exclusive. That is, this feature also facilitates the combination of data obtained from passive monitoring and online testing to have more data and perform a better analysis.

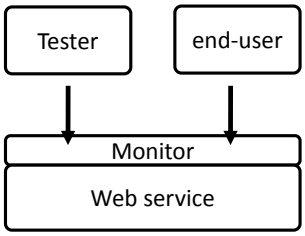

Fig. 4 Combination of passive monitoring and on-line testing

F3. Extensible with new quality attributes:

In a monitoring system, it is mandatory to provide the ability to compute new quality attributes as they are required (R1.2). To provide the extensibility with respect to new quality attributes, we provide two techniques, namely, at the conceptual and execution levels respectively (see Fig. 5).

At the conceptual level, quality characteristics, attributes and metrics based may be structured following some quality model for web services, e.g. the standard ISO/IEC 25010 (ISO/IEC, 2010) and aligned with the common terminology used. In (Oriol, Marco, & Franch, 2014a) we conducted a systematic mapping in order to identify the list of quality attributes and map them in the standard ISO/IEC 25010. By following this technique, new metrics can be added in a clearly structured manner following the standard, and interoperability and integration capabilities with other frameworks is also easier as they share a common framework of understanding.

At the execution level, SALMon is able to plug-in the required business logic that computes new metrics through a piece of software that we name measure instrument. A measure instrument is a component that is responsible to compute the values of a single metric. Hence, new metrics can be added by plugging the respective Measure Instruments. Adding new measure instruments can be done at runtime without the need of decommissioning or stopping the monitoring system.

These techniques also enable SALMon to monitor metrics at the infrastructure level (R3.1)

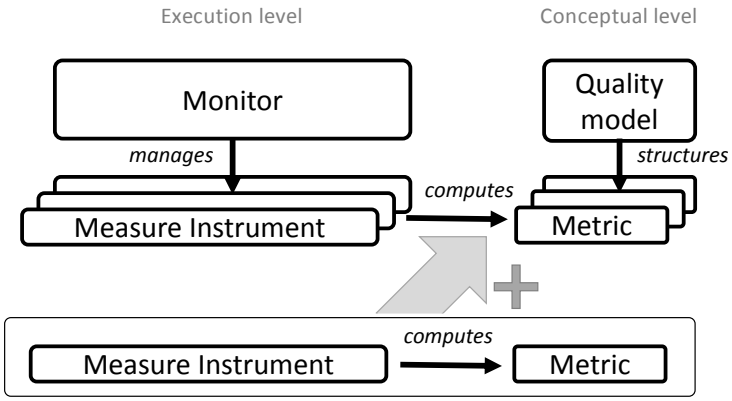

Fig. 5 Extensibility of metrics at the conceptual and execution level

F4. For any type of web service:

The high heterogeneity of the technologies used to implement web services requires a monitoring strategy not limited to a particular technical solution (R1.1). The monitor should not be attached to a particular technology and include the required modularity to extent it to support new kinds of web services. In contrast to other solutions, in our proposal, the messages are processed by the Measure Instruments, whereas the core of the monitor is agnostic over the technical specifications of the messages, leading the capability to monitor different type of web services (e.g. SOAP, RESTful services).

F5. Combines push and pull notification mechanisms:

Each activity requires different mechanisms to report the QoS data. In self-adaptive systems it is required to be notified as soon as possible with the updated QoS of the web services in order to perform an adaptation, whereas in service selection and deployment it is more convenient to provide the QoS of the web services only when they are required. To this aim, two strategies to retrieve the monitored QoS exist:

- Push strategy: QoS updates are notified to the subscribed components, which receive the data as soon as the web service is invoked and the quality attributes are computed.

- Pull strategy: QoS updates are requested by the components whenever they need it.

SALMon combines both notification mechanisms to provide either the monitored data on-demand or as soon as the monitored data is gathered.

F6: High interoperability:

Each SBS life-cycle activity requires specific features and specialized functionality that are implemented by dedicated software components. In order to be interoperable with these components (R1.4), the monitoring system must present a highly versatile architecture. To this aim, SALMon has been implemented following the SOA paradigm, where the different monitoring modules are web services, which facilitates the capability of each module to be easily coupled, replaced and decoupled in the monitoring system.

F7: High efficiency:

The monitor has to be efficient in reduce the consumption of resources (R3.2). SALMon has a modular architecture where the modules which are not required can be unplugged. Moreover SALMon has the capability to be deployed and decommissioned at runtime in order to reduce the consumption of resources.

4.2 SALMon's architecture

As mentioned above, SALMon has been designed following the SOA principles. These principles focus on high cohesion and low coupling aspects, which improves the reusability and maintainability of the software (Nitto, Sassen, Traverso, & Zwegers, 2009). To this aim, SALMon is composed of several modules, each one with the constituent services required to fulfil a specific activity:

- Core module: implements the features that are required along the different activities.
- MMD module: implements the model-based management system.
- Testing module: implements the online testing strategy.
- Subscription module: implements the push notification strategy.
- Data module: implements the storage repository of the monitored data.
- Monitor DB module: implements a facade to interact with the data module.

Fig. 6 shows these modules using a variation of the SAP-TAM notation (SAP, 2007).

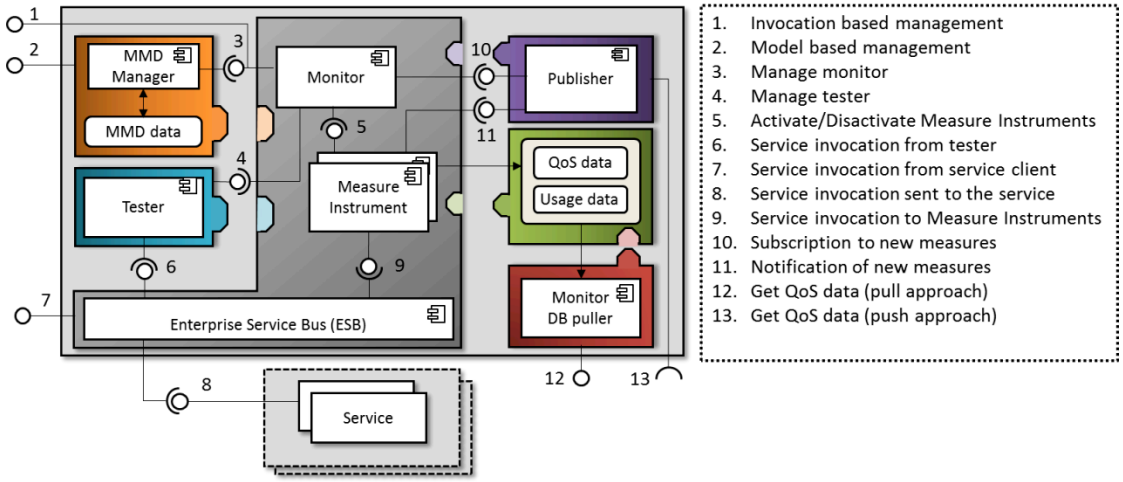

Fig. 6 SALMon's architecture

4.2.1 Core module

The *Core module* is the main module of the monitoring system and the only one which is mandatory. It consists of the components required along the different activities to correctly fulfil the monitoring process. These components are the *Monitor*, the *Measure Instruments* and the *ESB*.

Monitor: The *Monitor* is the main component of the system. It is a service that includes in its WSDL interface the capability to be managed directly, accomplishing the invocation-based strategy. The Monitor does not compute directly the values of the quality metrics; instead, the *Monitor* is responsible for managing the *Measure Instruments*. For each quality metric to evaluate, the *Monitor* activates the corresponding *Measure Instrument*. The *Monitor* also manages other (optional) modules, namely the *Tester* and *Publisher* modules, if present.

Measure Instrument: Each *Measure Instrument* includes the business logic to compute the value of a specific quality metric. A *Measure Instrument* can be either internal or external. An *Internal Measure Instrument* is included as a component of the *Core Module* and are the mostly wide used quality metrics (e.g. response time, availability, …), whereas *External Measure Instruments* are web services with a common interface that implement a specific quality metric required by the user (e.g. size of attached files). In such a manner, the *Monitor* is extensible with the addition of new metrics. The *Measure Instruments* receive the messages to monitor from the ESB.

ESB: In order to capture the messages from a service consumer, the service consumer must not invoke the service directly but through the usage on an *Enterprise Service Bus (ESB)*. This is fulfilled by specifying the target address using the standard WS-Addressing in the header of the messages and performing the invocation to the *ESB*. The *ESB* receives the message requests and forwards them to the actual web service. In order to avoid delays caused by message redirections, the *ESB* can be placed to the server or client side. The location of the *ESB* also depends on the metrics to calculate. For instance, to compute the response time, which includes the network delay, the *ESB* is required to be deployed near or at the client side. On the contrary, to compute the execution time, excluding the network delay, the *ESB* has to be deployed near or at the server side. The architecture allows deploying more than one *ESB*, which can be combined by applying the required redirections. In such a manner, we (1) allow the computation of server and client side metrics and (2) provide the ability to avoid bottlenecks, by instantiating new *ESBs* in case of high traffic. In parallel to invoking the web service, the *ESB* forwards the messages accompanied with their timestamps to the activated *Measure Instruments*, so they can compute the values of the quality metrics.

4.2.2 MMD module

This module is used to implement the model-based management system. In this regard, it is required a model transformation from the source model to the model that configures the monitor. The source model is not limited to SLA documents, but also can function with other types of models (e.g. goal-based models). By specifying the appropriate transformation rules and using techniques as those presented in (Krzysztof Czarnecki, 2003), we are able to derive from the source model to the model that configures the monitor (i.e. the target model). With respect to the target model, we propose to use the MMD (Muller et al., 2013), a specification which includes what is required to monitor and how the data is to be gathered. The MMD, as a standalone model, is a formal XML document whose main functions are (1) the specification of a monitoring configuration to gather properly the required metrics and (2) a container to store and retrieve the monitoring results in the same document structure. In this sense, the MMD is used as both input and output model of the system. An excerpt of an MMD with monitoring results is depicted in Fig. 7. Details of the MMD structure can be found at (Muller et al., 2013).

```
<MonitoringManagementDocument>
 ...
 <serviceMetric>
  <metric>AverageAvailability</...>
  <localPeriodInit>2013-05-18T18:02:38</...>
  <localPeriodEnd>2014-01-01T00:00:00</...>
   <measure>
    <value>100</value>
    <timeStamp>2013-05-18T18:02:38</timeStamp>
   </measure>
 </...>
 ...
 <operationMetric opName="explainNonCompliance">
  <metric>AverageResponseTime</...>
  <localPeriodInit>2013-05-18T18:02:38</...>
  <localPeriodEnd>2014-01-01T00:00:00</...>
   <measure>
    <value>3421</value>
    <timeStamp>2013-05-18T18:02:38</timeStamp>
   </measure>
 </...>
 ...
</MonitoringManagmentDocument>
```

Fig. 7: Excerpt of an MMD with monitoring results.

MMD Manager: the *MMD Manager* is responsible for managing the MMD. It also invokes the *Monitor* accordingly to the *MMD* rules and updates the document with the monitoring results.

MMD data: The MMDs are stored and updated in a secured repository.

4.2.3 Testing module

The testing module is used to perform online testing to the target web services. It consists of the *Tester Engine*

Tester Engine: The *Tester Engine* is activated by the monitor by specifying the target web services, the operations to invoke, the input to perform the tests and the time interval between invocations. The tester uses exactly the same infrastructure that is used for the messages from a real service consumer. The test messages goes through the same ESB redirections. In such a manner, any duplicity is avoided, and the combination of both approaches is assured. To identify that these requests do not belong to a real user but are from the *Tester Engine*, the test messages include an identifier tag in the header of the message.

4.2.4 Subscription module

The subscription module implements the push notification strategy to inform updates regarding the monitored web services as soon as there are gathered. It is operationalized by the *Publisher* service.

Publisher: The *Publisher* implements the observer pattern for web services as defined in [ref]. The Monitor subscribes the metrics or other data computed by the *Measure Instruments* to the interested party (e.g. the consumer, provider, an adaptation component, etc.). The *Publisher* handles the list of subscribed parties and their subscriptions and notifies the events whenever a new value of interest is retrieved by the *Measure Instruments*. The subscribed parties must implement a web service interface that handles such notifications.

4.2.5 Data module

The Data module is a repository to store the monitored data, which can be accessed internally by the *Monitor* or externally by the *Monitor DB puller*. It consists of two major building blocks:

QoS data: Is the part of the repository which stores all the monitored QoS information (e.g. response time, availability, etc.)

Usage data: is the part of the repository which stores all the information related to the usage of the web services (e.g. invocations performed, inputs used, etc.)

The current repository has been implemented as a MySQL database, but other technologies could be used as required by the SBS.

4.2.6 Monitor DB module

This module is used to get the information from the Data module externally from the monitor. It consists of the *Monitor DB puller*.

Monitor DB puller: It provides a WSDL interface to access the data from the repository, decoupling the user to a particular storage technology.

5. SALMon validation in the lifecycle of an SBS

SALMon has been validated in depth in the different scenarios supporting the previously mentioned stages of the lifecycle of an SBS. We detail here, how SALMon has been used in the different frameworks of several research groups to accomplish a varied set of activities. The features that were implemented and the results of such collaborations are detailed.

5.1 Service selection

The selection framework

SALMon has been integrated and validated with a particular web service selection framework to demonstrate the feasibility of the monitoring framework in this activity. Particularly, SALMon has been integrated within WeSSQoS (Cabrera et al., 2011)(Cabrera et al., 2009). WeSSQoS is a configurable quality-aware web service selection framework that combines multiple web service repositories and algorithms to provide and augmented user experience in the selection of web services. The repositories integrated with WeSSQoS provide information about the web services and some statically defined quality attributes. WeSSQoS combines all this information with SALMon for an accurate and up-to-date QoS data of the web services (see http://gessi.lsi.upc.edu/wessqos).

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Tester* and the *Data* module (see Fig. 8).

To monitor these web services, the *Tester* component of SALMon performs the tests over the web services registered in the repository of WeSSQoS. The QoS data is then stored in the *QoS data* module, which can ultimately be retrieved by the *Selector* module of WessQoS.

The *Selector* module merges all the data and executes the normalization and ranking algorithms provided by the Normalize & Ranking Module, resulting in a ranked list of the discovered web services that better fulfils the user's needs.

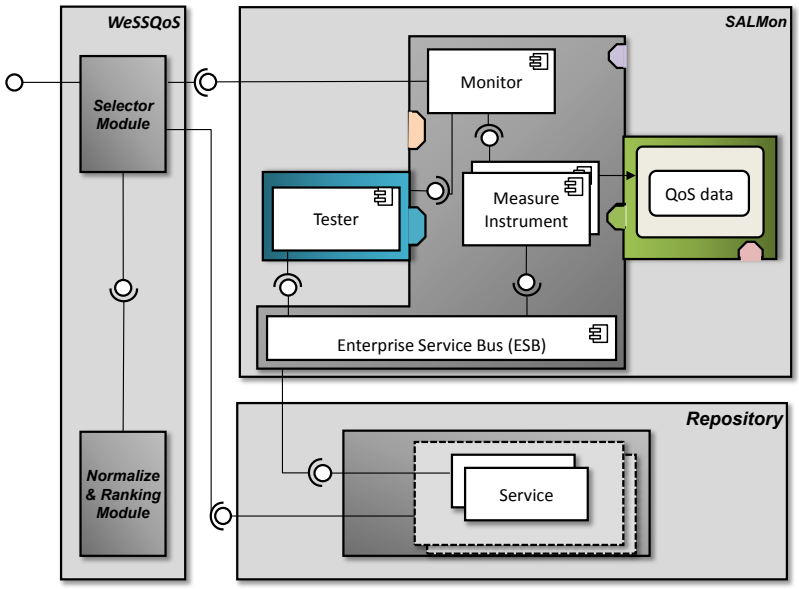

Fig. 8 SALMon in service selection

5.2 SBS deployment

SBS deployment framework

To demonstrate the feasibility of SALMon during the SBS deployment activity on heterogeneous deployment infrastructures, SALMon has been integrated and validated within a cloud federation system. Particularly, in the Federated Cloud Management (FCM) framework (Marosi et al., 2011) (Kertesz et al., 2012b). FCM is a brokering system that integrates heterogeneous cloud systems and provides a unique broker to deploy and use SBS in the cloud seamlessly. SALMon has been integrated with this solution to monitor the QoS at the infrastructure level.

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Tester*, the *Data* module and the *Monitor DB puller* module. Moreover, an external web service designed for the framework, named *M3S,* is also required (see Fig. 9).

M3S is a web service that incorporates several methods designed to make usage of the infrastructure of the platform where it has been deployed (e.g. network, CPU, etc.). By monitoring the performance of the different methods provided by the web service, the QoS at the infrastructure level is computed.

To be accurate in the results of monitoring *M3S*, the core of SALMon and the *Tester* are deployed in the same platform as the *M3S* service. The *Data* module and the *Monitor DB puller* are deployed outside of the aforementioned platform in order to (1) avoid the consumption of memory and disk in the platform that is being monitored and (2) be able to access the data even when the core of SALMon and the *M3S* have been decommissioned. The Federated Cloud Management System is then able to retrieve the QoS through the *Monitor DB puller*.

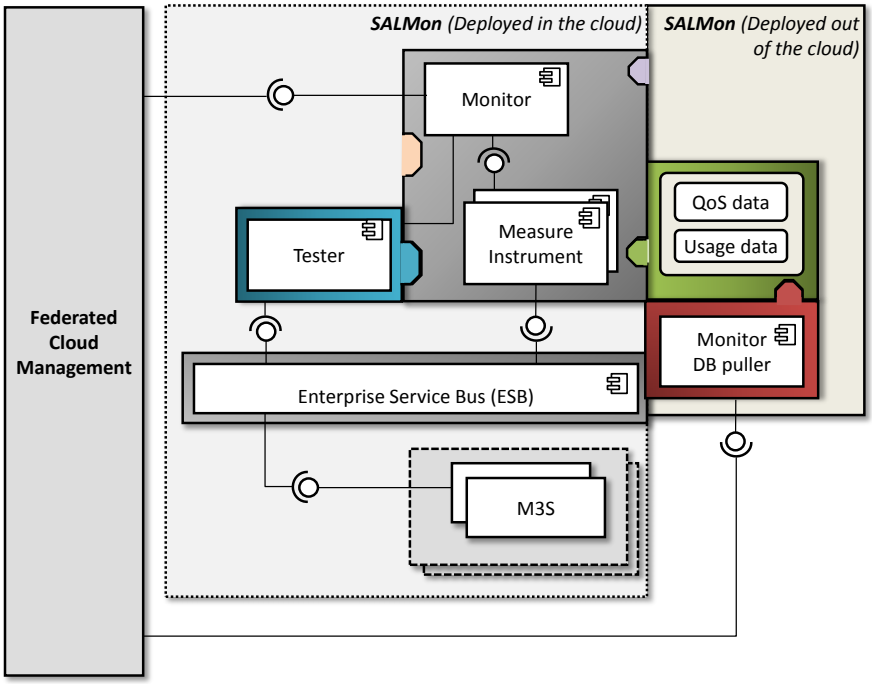

Fig. 9. SALMon in SBS deployment

5.3 Quality assessment

Quality assessment framework

The QoS to be achieved during the execution of a web service is usually agreed in the form of an SLA. SALMon has been integrated and validated within an SLA analysis platform called ADA (Müller, Resinas, & Ruiz-Cortés, 2009), and extended to form a new technological solution named SALMonADA (Muller et al., 2012) (Muller et al., 2013). SALMonADA is a framework able to automatically monitor, detect and analize violations of SLA clauses during the execution of the web services. Beyond other features, SALMonADA is able to provide human readable explanations of SLA violations for highly expressive SLAs as soon as they occur (see http://www.isa.us.es/ada.source/SLAnalyzer/).

Required components of SALMon and usage

The required components of SALMon in this activity are: *MMD* module, *Subscription* module and *Data* module (see Fig. 10).

To perform the analysis, SALMonADA automatically generates, from the SLA, the MMD document to configure the monitor. As the service client invokes the web service through the *ESB*, the web services are being monitored and every new computed metric is notified through the *Subscription* module to (1) generate the updated MMD with the monitored metrics and (2) compute the analysis if any violation has occurred.

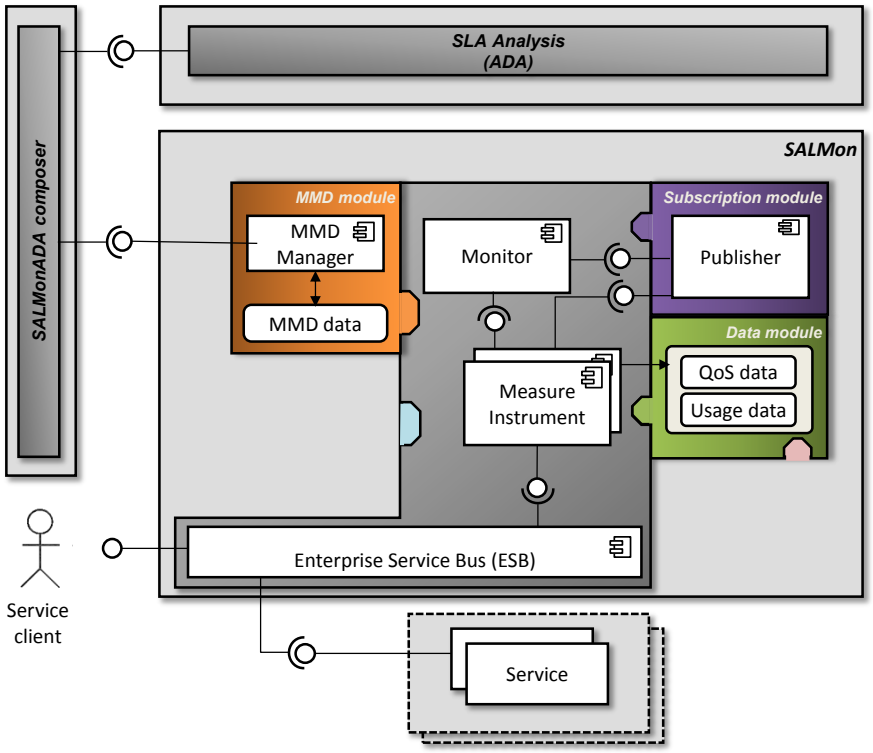

Fig. 10. SALMon in quality assessment

5.4 SBS adaptation

SBS adaptation frameworks

SALMon has been integrated and validated with different frameworks for self-adaptive systems: MAESoS (Franch et al., 2011), PROSA (Sammodi et al., 2011), PROTEUS (Schmieders et al., 2011) and CARE (Oriol et al., 2012).

- MAESoS (Franch et al., 2011) is a framework that supports self-adaptation of SBS by combining several models. It uses *i** models to specify requirements, tasks and dependencies; feature models to specify rules and alternatives; and quality models for the description of the QoS required.
- PROSA (Sammodi et al., 2011) is a framework that supports self-adaptation of SBS based on failure prediction. This is achieved by combining passive monitoring and active testing dynamically.
- PROTEUS (Schmieders et al., 2011) is a framework that supports self-adaptation of SBS to prevent violations due to malfunction of the executed services. The adaptation strategies are focused on mitigating the effects of any service malfunction.
- CARE (Oriol et al., 2012) is a framework that supports the adaptation of requirements in SBS by involving the end-users, if needed, to satisfy their needs.

Required components of SALMon and usage

The required components of SALMon in this activity are: the *Subscription* module and *Data* module. It also requires a new component, which is an *Analyser* to detect the adaptation needs (see Fig. 11).

The *Analyser* is able to check simple conditions and trigger an adaptation need if such conditions are not met. The *Analyser* configures the monitor either in passive monitoring or online testing strategies, and hence, both proactive and reactive adaptations are supported.

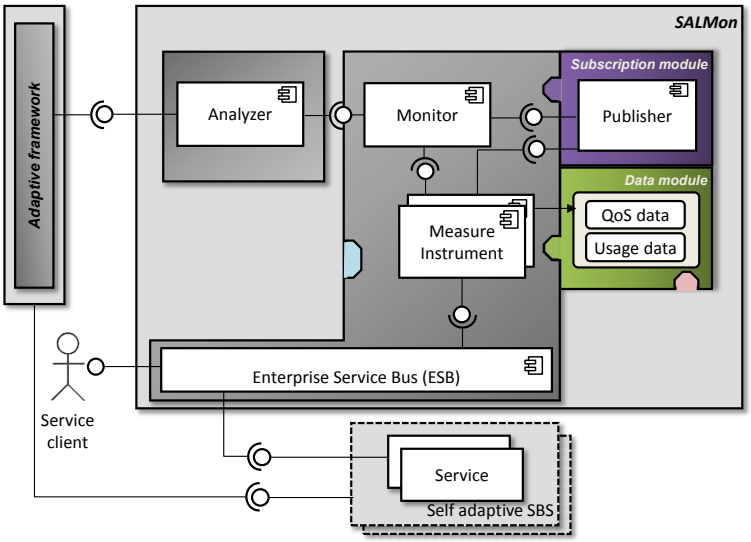

Fig. 11. SALMon in SBS adaptation

6. Performance evaluation

In this section we evaluate the performance of SALMon. Particularly, we aim at evaluating the overhead and the capacity of SALMon. To quantify it, we evaluate by means of an adequate benchmark: (1) the response time overhead under normal operating conditions (i.e. one invocation at a time), and (2) the maximum throughput SALMon is able to handle without incrementing such overhead.

To perform the evaluation, we invoke a set of real services, and compare the response time by invoking the services both directly and through SALMon. To obtain a set of representative services, we started from a list of 393 services available in the public repository[4] developed by XMethods, an organization dedicated to promoting the development, deployment, and use of web services. Then we applied the following criteria: (1) We first considered the most recently submitted services under the assumption that recent services are more likely to be available than older services. Considering the length of the list, we established as threshold the 1/3 of the complete list. (2) From the resulting 131 services, we removed those ones falling into any of the following situations: were not available, were payment services, required registration or didn't have stateless operations, resulting in 23 services. (3) We tested these 23 services and removed those ones that had errors in their descriptions (WSDL), or that gave faulty results in their functionality when invoked, resulting in a final list of 11 services from 8 different service providers, deployed on their respective servers. The list of services and their WSDLs are available at (Oriol, Marco, & Franch, 2014b).

[4] http://www.xmethods.net/ve2/Directory.po

6.1 Overhead evaluation

The ESB Apache Synapse included in SALMon adds a low overhead while handling the HTTP messages. The ESB has a non-blocking HTTP transport and multithreaded mediation, which as we measured, results in a negligible 1-3 ms. overhead. Nevertheless, in our approach, there are three possible locations where SALMon can be deployed: at the server side, at the client side, or in an intermediate server (i.e., in the middle). Depending on the location, the overhead experienced by the consumer varies. If SALMon is placed at the server or client side, there is an overhead on the resources due to the execution of the monitoring components. However, this overhead can be easily compensated by adding more resources.

If SALMon is placed in the middle, it does not produce an overhead on the resources of the client or server side. However, the deployment of SALMon in an intermediate server adds a network delay from Internet Service Providers due to the redirection of the messages. To evaluate this overhead, we executed each of the selected services 100 times, with a throughput of 1 invocation per second. One key issue regarding the analysis of the results is dealing with outliers (e.g. network failures that increase the response time of an invocation). Commonly used methods to deal with outliers require that the data follow a Gaussian distribution (Hawkins, 1980). However, from the experiment results we have observed that response times do not follow a Gaussian distribution, but an exponentially modified Gaussian or inverse Gaussian distribution (See Fig. 12).

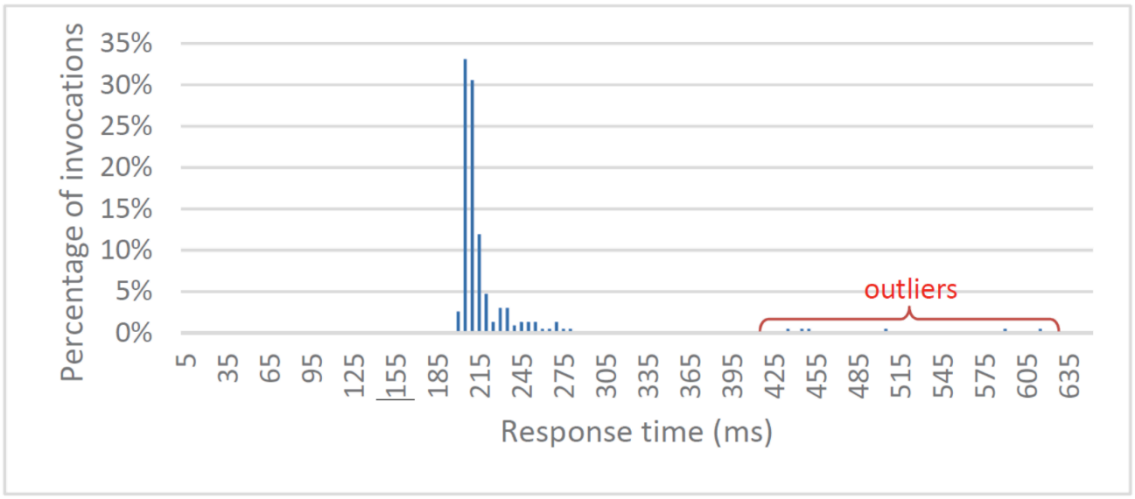

Fig. 12 Response time distribution of one of the monitored web services

As shown, the population grows rapidly on the left-hand side and decreases slowly on the right-hand side in the form of a tail. Those elements that are far away from the mean are considered outliers. To deal with these outliers, we followed the methods described and evaluated by Ratcliff for dealing with response time outliers (Ratcliff, 1993). Although Ratcliff studied response time of people in the field of psychology, the results can be applied to any model that follows the inverse Gaussian distribution. According to Ratcliff, we will not compute directly the average response time (which is not a robust estimator in front of outliers), but we will use two other robust estimators, namely, the inverse transformation and removing outliers at a standard deviation distance. The first estimator consists on applying the inverse response time (1/R) on each individual invocation, calculate the average, and then invert the result. The second estimator consists on calculating the average response time after removing the outliers at a standard deviation distance. We computed these methods over the invocations on each

service for both directed and redirected invocations. As a result, we got two robust estimators per each service. We applied these estimators to the response time of direct and redirected invocations in order to calculate the response time overhead introduced in the web service by the deployment of SALMon. We decided to relate the two parameters with a linear interpolation curve fitting method with the aim of obtaining mathematical functions approximating the response time overhead.

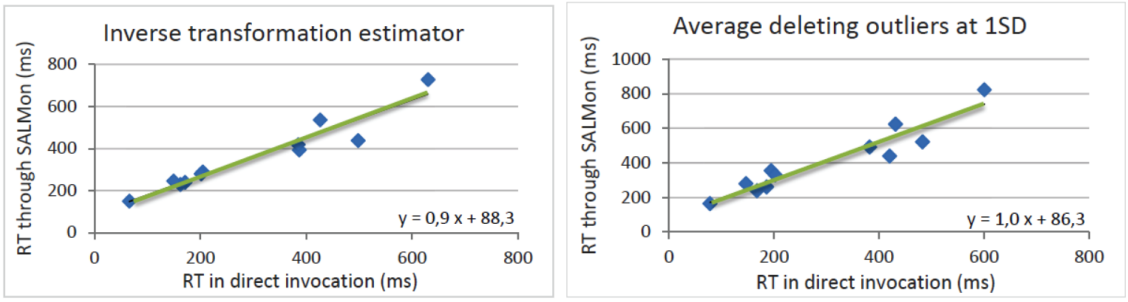

Fig. 13 Response time of the web services, invoked directly and through SALMon

Fig. 13 shows the obtained functions for each of the two applied robust estimators methods, which are: y = 0.9x + 88.3ms and y = 1.0x + 86.3ms. These results show that the overhead of SALMon is a constant value between 86 and 89 ms. It is worth to mention that some deployment strategies can be applied to mitigate any concern. We argue that for web services that require extremely fast response times and need to avoid the 86-89 ms. overhead, SALMon can be deployed at the client or server side, as the overhead is mainly caused by the network delay. For other types of web services, we argue that a deployment in the middle is preferred, since this solution is less intrusive to both the client and the provider server, as it does not require the installation of the monitor in their infrastructures.

6.2 Capacity evaluation

To evaluate the capacity of SALMon, we invoked each web service directly and through SALMon with different throughputs. We tested the list of web services with throughputs from 1 invocation per second up to 50 invocations per second, and per each throughput we made 100 invocations.

For web services with a capacity lower than SALMon, the results are not useful to identify the capacity of SALMon. For web services with a high capacity, e.g. CalcService, we identified that SALMon is able to operate correctly with throughputs up to 30 invocations per second. For throughputs that go beyond 30 invocations per second, the capacity of SALMon is outreached after several invocations (usually taking more than 100 invocations). An excerpt of the results is shown in Fig. 14; the complete list of results (graphical and raw data) is available at (Oriol, Marco, et al., 2014b).

SALMon can monitor web services with a throughput up to 30 invocations per second. Nevertheless, for web service that require a higher throughput, such limitations can be overcome by deploying more ESBs in a distributed set of servers.

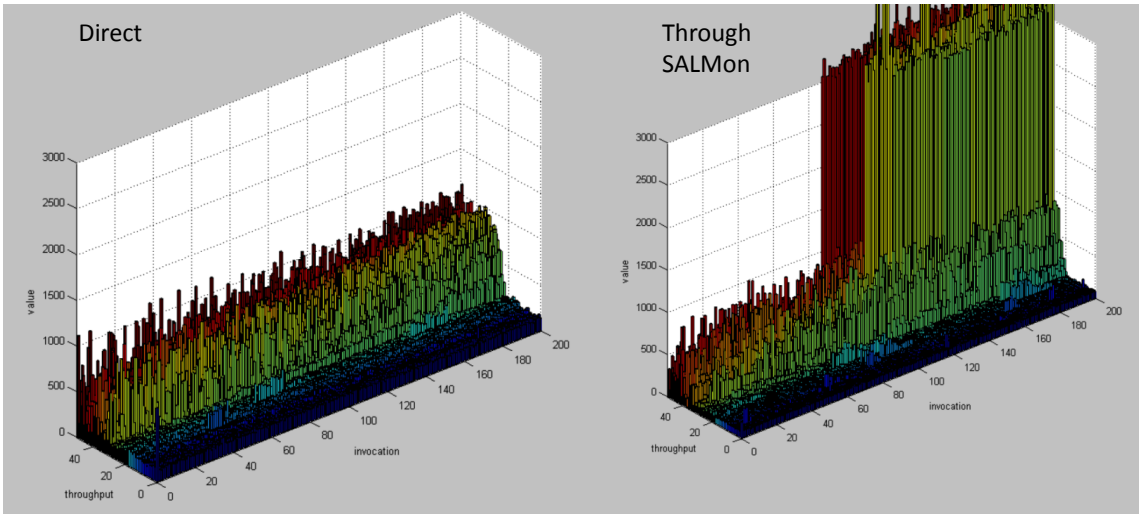

Fig. 14 Response times of CalcService with different throughput (direct and through SALMon)

7. Conclusions

In this paper, we have presented SALMon, a highly versatile QoS monitoring framework able to support the whole SBS lifecycle. We have first identified the requirements of the different activities in the lifecycle by (1) analysing the needs as described in the literature, and (2) conducting meetings and interviews with members of different research groups who work on these activities. Then, from the elicited requirements, we have designed a set of features and developed the SALMon framework. The main advantages that SALMon offers with respect to other existing frameworks, which have been summarized in Table 3, are:

- Combination of model-based and invocation-based strategies
- Combination of passive monitoring and online testing strategies
- Extensibility with new quality attributes
- Low coupling to the monitored services' technology
- High interoperability

According to our literature review performed over 24 world-leading journals and conferences on the topic in the interval 2008-2014, existing approaches do not cover the different phases of the SBS lifecycle. In more quantitative terms, the approach covering the largest number of the 13 requirements we have identified and implemented in SALMon, covers 9 of them (Psiuk, Bujok, & Zieliski, 2012). Also the validation of the proposed approaches is limited, with SBS adaptation being the most validated phase (7 approaches out of 14).

In contrast, SALMon is a monitoring framework that supports all the different activities in the SBS lifecycle by providing the knowledge base (accurate and complete QoS) to several expert systems. As described, SALMon has been integrated with the following frameworks:

- WeSSQoS (Service selection): An expert system capable of dealing with several pluggable strategies to perform the selection of web services according to the requirements of the user.
- FCM (SBS deployment): An expert system capable of deciding the best deployment strategy on a cloud federation system.
- SALMonADA (Quality assessment): An expert system capable of identifying and reporting an SLA violation and its cause.

- MAESoS, PROSA, PROTEUS, CARE (Adaptation): Expert systems capable of performing an adaptation whenever malfunctions in the system occur.

The monitoring requirements for every framework are very diverse, and in some cases, may apparently be in conflict among each other. This lesson learned has brought us to the design of a modular system, capable of plugging in the different modules in such a manner that they can co-exist together.

A limitation of SALMon is that the reconfiguration of the monitor is currently performed by the SBS administrator (i.e. if there is an adaptation on the SBS, SALMon should be reconfigured accordingly). Another aspect that has been out of the scope is the security of the system (e.g. against DDoS attacks).

The future work spreads over several directions. First, to overcome current limitations, we aim at using configuration management techniques to automatically reconfigure the monitor according to SBS adaptations. Secondly, we envisage the use of advanced techniques for supporting the decision on when to adopt passive monitoring or online testing as strategy to get QoS from monitored services. This is not a straightforward decision since different parameters as efficiency, accuracy of results and invasiveness need to be reconciled. A third line of future research emphasises the connection between the monitor and adaptation decisions. Given that adaptation has a cost, the decision of adapting at runtime needs to be carefully assessed in order to avoid false positives (i.e., to avoid adapting when at the end it would have been not necessary). A way to improve the accuracy of the decision is to increase the amount and quality of the monitored data, but since also this increase has its own cost, techniques are needed to find the optimal compromise between both factors. Finally, we plan to extend the testing module of SALMon to define test cases using a standard model (e.g. UML Testing Profile) and implement a repository of Measure Instruments that gathers the quality metrics.

8. Acknowledgments

This work has been supported by the Spanish Government under the CICYT projects TIN2010-19130-C02-01 and TIN2013-44641-P and by the S-Cube Network of Excellence funded by the EU, contract n. 215483.

9. References

Alhamazani, K., Ranjan, R., Mitra, K., Jayaraman, P. P., Huang, Z., Wang, L., & Rabhi, F. (2014). CLAMS: Cross-layer Multi-cloud Application Monitoring-as-a-Service Framework. In *2014 IEEE International Conference on Services Computing* (pp. 283–290). IEEE. doi:10.1109/SCC.2014.45

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., … Xu, M. (2004). Web services agreement specification (WS-Agreement). In *Global Grid Forum*.

Andrikopoulos, V., Bertoli, P., & Bindelli, S. (2008). State of the art report on software engineering design knowledge and survey of HCI and contextual knowledge. *Project Deliverable PO-JRA- …*.

Baresi, L., & Guinea, S. (2011). Self-Supervising BPEL Processes. *IEEE Transactions on Software Engineering*, *37*(2), 247–263. doi:10.1109/TSE.2010.37

Baresi, L., Guinea, S., Nano, O., & Spanoudakis, G. (2010). Comprehensive Monitoring of BPEL Processes. *IEEE Internet Computing*, *14*(3), 50–57. doi:10.1109/MIC.2010.66

Baresi, L., Guinea, S., Pistore, M., & Trainotti, M. (2009). Dynamo + Astro: An Integrated Approach for BPEL Monitoring. In *IEEE International Conference on Web Services (ICWS)* (pp. 230–237). doi:10.1109/ICWS.2009.67

Benharref, A., Dssouli, R., Serhani, M. A., & Glitho, R. (2009). Efficient traces' collection mechanisms for passive testing of Web Services. *Information and Software Technology*, *51*(2), 362–374. doi:10.1016/j.infsof.2008.04.007

Bieberstein, N., Laird, R., Jones, K., & Mitra, T. (2008). *Executing SOA: A Practical Guide for the Service-Oriented Architect* (1st ed.). Addison-Wesley Professional.

Burgstaller, B., Dhungana, D., Franch, X., Grunbacher, P., López, L., Marco, J., … Stockhammer, R. (2009). Monitoring and Adaptation of Service-oriented Systems with Goal and Variability Models. In *LSI, Technical Report.*

Cabrera, O., Oriol, M., Franch, X., López, L., Marco, J., Fragoso, O., & Santaolaya, R. (2011). WeSSQoS: A Configurable SOA System for Quality-aware Web Service Selection. In *arXiv, Technical Report*.

Cabrera, O., Oriol, M., López, L., Franch, X., Marco, J., Fragoso, O., & Santaolaya, R. (2009). WeSSQoS: Un sistema SOA para la selección de servicios web según su calidad. In *Proceeding of Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB)* (pp. 76–89).

Comuzzi, M., Kotsokalis, C., Spanoudakis, G., & Yahyapour, R. (2009). Establishing and Monitoring SLAs in Complex Service Based Systems. In *IEEE International Conference on Web Services (ICWS)* (pp. 783–790). Ieee. doi:10.1109/ICWS.2009.47

Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., & Pohl, K. (2008). A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, *15*(3-4), 313–341. doi:10.1007/s10515-008-0032-x

Erl, T. (2008). *Soa: principles of service design* (Vol. 1). Prentice Hall Upper Saddle River.

Fei, L., Fangchun, Y., Kai, S., & Sen, S. (2008). A Policy-Driven Distributed Framework for Monitoring Quality of Web Services. In *IEEE International Conference on Web Services (ICWS)* (pp. 708–715). Ieee. doi:10.1109/ICWS.2008.123

Franch, X., Grunbacher, P., Oriol, M., Burgstaller, B., Dhungana, D., Lopez, L., … Pimentel, J. (2011). Goal-Driven Adaptation of Service-Based Systems from Runtime Monitoring Data. In *35th Annual Computer Software and Applications Conference Workshops* (pp. 458–463). IEEE. doi:10.1109/COMPSACW.2011.83

Gentzsch, W., Grandinetti, L., Joubert, G., Ricci, L., Baraglia, R., Lucas-Simarro, J. L., … Llorente, I. M. (2013). Scheduling strategies for optimal service deployment across multiple clouds. *Future Generation Computer Systems*, *29*(6), 1431–1441.

Gu, Q., & Lago, P. (2007). A stakeholder-driven service life cycle model for SOA. In *2nd International Workshop on Service oriented software engineering in conjunction with the 6th ESEC/FSE joint meeting - IW-SOSWE '07* (pp. 1–7). New York, New York, USA: ACM Press. doi:10.1145/1294928.1294930

Guinea, S., & Kecskemeti, G. (2011). Multi-layered monitoring and adaptation. *9th International Conference on Service-Oriented Computing (ICSOC)*, 359–373.

Hawkins, D. M. (1980). *Identification of Outliers* (1st ed.). Dordrecht: Springer Netherlands. doi:10.1007/978-94-015-3994-4

ISO/IEC. (2010). *ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. ISO/IEC.

Kang, G., Liu, J., Tang, M., Liu, X. (Frank), & Fletcher, K. K. (2011). Web Service Selection for Resolving Conflicting Service Requests. In *IEEE International Conference on Web Services (ICWS)* (pp. 387–394). IEEE. doi:10.1109/ICWS.2011.37

Katsaros, G., Kübert, R., & Gallizo, G. (2011). Building a Service-Oriented Monitoring Framework with REST and Nagios. In *IEEE International Conference on Services Computing* (pp. 426–431). Ieee. doi:10.1109/SCC.2011.53

Kertesz, A., Kecskemeti, G., Marosi, A., Oriol, M., Franch, X., & Marco, J. (2012a). Integrated Monitoring Approach for Seamless Service Provisioning in Federated Clouds. In *20th Euromicro International Conference on Parallel, Distributed and Network-based Processing* (pp. 567–574). IEEE. doi:10.1109/PDP.2012.25

Kertesz, A., Kecskemeti, G., Oriol, M., Kotcauer, P., Acs, S., Rodríguez, M., … Franch, X. (2013). Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach. *Journal of Grid Computing*, *11*(4), 699–720. doi:10.1007/s10723-013-9269-0

Kertesz et al. (2012b). A holistic service provisioning solution for Federated Cloud infrastructures. In *1st International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube)* (pp. 25–26). IEEE. doi:10.1109/S-Cube.2012.6225504

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering, Version 2.3*. *Version* (Vol. 2).

Krzysztof Czarnecki, S. H. (2003). Classification of Model Transformation Approaches. In *2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA* (pp. 1–17).

Lin, K.-J., Panahi, M., Zhang, Y., Zhang, J., & Chang, S.-H. (2009). Building Accountability Middleware to Support Dependable SOA. *IEEE Internet Computing*, *13*(2), 16–25. doi:10.1109/MIC.2009.28

Mahbub, K., Spanoudakis, G., & Zisman, A. (2010). A monitoring approach for runtime service discovery. *Automated Software Engineering*, *18*(2), 117–161. doi:10.1007/s10515-010-0077-5

Marosi, A. C., Kecskemeti, G., Kertesz, A., & Kacsuk, P. (2011). FCM: an Architecture for Integrating IaaS Cloud Systems. In *2nd International Conference on Cloud Computing, GRIDs, and Virtualization* (pp. 7–12).

Michell Smith, D., Petri, G., Natis, Y. V., Scott, D., Warrilow, M., Heiser, J., … Lheureux, B. J. (2013). *Predicts 2014: Cloud Computing Affects All Aspects of IT*. Gartner document.

Michlmayr, A., Rosenberg, F., Leitner, P., & Dustdar, S. (2010). End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. *IEEE Transactions on Services Computing*, *3*(3), 193–205. doi:10.1109/TSC.2010.20

Moser, O., Rosenberg, F., & Dustdar, S. (2008). Non-intrusive monitoring and service adaptation for WS-BPEL. *Proceeding of the 17th International Conference on World Wide Web - WWW '08*, 815. doi:10.1145/1367497.1367607

Moser, O., Rosenberg, F., & Dustdar, S. (2010). Event Driven Monitoring for Service Composition Infrastructures. In *Web Information Systems Engineering – WISE 2010* (pp. 38–51).

Moser, O., Rosenberg, F., & Dustdar, S. (2012). Domain-Specific Service Selection for Composite Services. *Software Engineering, IEEE Transactions on*, *38*(4), 828–843.

Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Cortes, A., & Rodriguez, M. (2013). Comprehensive Explanation of SLA Violations at Runtime. *IEEE Transactions on Services Computing*, *7*(2), 168–193. doi:10.1109/TSC.2013.45

Muller, C., Oriol, M., Rodriguez, M., Franch, X., Marco, J., Resinas, M., & Ruiz-Cortes, A. (2012). SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents. In *4th International Workshop on Principles of Engineering Service-Oriented Systems (PESOS)* (pp. 43–49). IEEE. doi:10.1109/PESOS.2012.6225938

Müller, C., Resinas, M., & Ruiz-Cortés, A. (2009). Explaining the Non-compliance between Templates and Agreement Offers in WS-Agreement. In *7th International Joint Conference ICSOC-ServiceWave* (pp. 237–252). Berlin, Heidelberg. doi:10.1007/978-3-642-10383-4

Nitto, E. Di, Sassen, A.-M., Traverso, P., & Zwegers, A. (2009). *At Your Service: Service-oriented Computing from an EU Perspective* (1st ed.). MIT Press.

Oriol, M., Franch, X., & Marco, J. (2014). Details on SALMon - State of the Art. Retrieved from http://gessi.lsi.upc.edu/salmon/slr

Oriol, M., Marco, J., & Franch, X. (2014a). Quality models for web services: A systematic mapping. *Information and Software Technology*, *56*(10), 1167–1182. doi:10.1016/j.infsof.2014.03.012

Oriol, M., Marco, J., & Franch, X. (2014b). SALMon evaluation. Retrieved from http://gessi.upc.edu/salmon/evaluation

Oriol, M., Qureshi, N. A., Franch, X., Perini, A., & Marco, J. (2012). Requirements Monitoring for adaptive service-based applications. In *Requirements Engineering: Foundation for Software Quality (REFSQ* (pp. 280–287). doi:10.1007/978-3-642-28714-5

Ortiz, G., & Bordbar, B. (2009). Aspect-Oriented Quality of Service for Web Services: A Model-Driven Approach. In *IEEE International Conference on Web Services* (pp. 559–566). Ieee. doi:10.1109/ICWS.2009.20

Papazoglou, M. P., Pohl, K., Parkin, M., & Metzger, A. (Eds.). (2010). *Service Research Challenges and Solutions for the Future Internet*. *S-CUBE Book* (Vol. 6500). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-17599-2

Psiuk, M., Bujok, T., & Zieliski, K. (2012). Enterprise Service Bus Monitoring Framework for SOA Systems. *IEEE Transactions on Services Computing*, *5*(3), 450–466. doi:10.1109/TSC.2011.32

Raimondi, F., Skene, J., & Emmerich, W. (2008). Efficient online monitoring of web-service SLAs. *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering - SIGSOFT '08/FSE-16*, 170–180. doi:10.1145/1453101.1453125

Ratcliff, R. (1993). Methods for dealing with reaction time outliers. *Psychological Bulletin*, *114*(3), 510–532. doi:10.1037/0033-2909.114.3.510

Sammodi, O., Metzger, A., Franch, X., Oriol, M., Marco, J., & Pohl, K. (2011). Usage-Based Online Testing for Proactive Adaptation of Service-Based Applications. In *35th Annual Computer Software and Applications Conference* (pp. 582–587). IEEE. doi:10.1109/COMPSAC.2011.81

SAP. (2007). Standardized Technical Architecture Modeling, Conceptual and Design Level. Retrieved January 27, 2014, from http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf

Schmieders, E., Micsik, A., & Oriol, M. (2011). Combining SLA prediction and cross layer adaptation for preventing SLA violations. In *2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services* (pp. 1–6).

S-Cube. (2009). Deliverable PO-JRA-1.2.1, State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs.

Shu, Z., & Meina, S. (2010). An architecture design of life cycle based SLA management. In *12th International Conference on Advanced Communication Technology (ICACT)* (pp. 1351–1355).

Wang, Q., Shao, J., Deng, F., Liu, Y., Li, M., Han, J., & Mei, H. (2009). An Online Monitoring Approach for Web Service Requirements. *IEEE Transactions on Services Computing*, *2*(4), 338–351. doi:10.1109/TSC.2009.22

Wang, S., Sun, Q., & Yang, F. (2012). Quality of service measure approach of web service for service selection. *IET Software*, *6*(2), 148. doi:10.1049/iet-sen.2010.0093

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. doi:10.1007/s13174-010-0007-6

Zheng, Z., Zhang, Y., & Lyu, M. R. (2010). Distributed QoS Evaluation for Real-World Web Services. In *IEEE International Conference on Web Services* (pp. 83–90). IEEE. doi:10.1109/ICWS.2010.10